

What is Claimed:

1. A method of providing an application layer access to a fixed memory address space of a device, the method comprising:
 - 5 constructing an object having elements which occupy said fixed memory address space;
whereby the application is provided access to the fixed memory address space directly through said object.
- 10 2. The method according to claim 1 further comprising:
identifying the fixed memory address space to be a hardware peripheral's memory mapped registers.
3. The method according to claim 1 further comprising:
 - 15 defining a class having base address and length parameters, which is used in constructing said object.
4. The method according to claim 1 further comprising:
defining a Java class having type, base address and
20 length parameters, which is used in constructing the object.
5. The method according to claim 1 wherein constructing the object comprises:
creating an object descriptor;
 - 25 creating an object handle for the object which points to the object descriptor.
6. The method according to claim 5 further comprising:
defining a Java class having base address, length and
30 type parameters, which is used in constructing the object.

7. The method according to claim 4 further comprising defining a new class having a class name <class name> as follows:

<class name> (base, length)

5 where <class name> is the name assigned to the new class, base is a parameter which specifies a type of object, base is a parameter which specifies a beginning address, and length is a parameter specifying a number of elements in the object, which when constructed, generates an object descriptor specifying
10 base, length, and a generates a handle which points to the object descriptor.

8. The method according to claim 7 further comprising generating an object descriptor specifying a default type.

15

9. The method according to claim 6 wherein new class is substantially defined in pseudocode as follows:

Class AnchoredArray

```
{  
20 public int element[ ];  
public AnchoredArray(int baseAddress, int length)  
{  
    element = lockDownElements(baseAddress,length);  
    Static private native int[ ] lockDownElements(int baseAddress, int length);  
25 }
```

```
void AnchoredArray_lockDownelements()  
{  
    int base = popStack();  
30 int length = popStack();  
    int *handle = malloc(SIZE_OF_HEADER);  
    (instance*) handle-> type= DEFAULT_TYPE  
    (arrayStruct*) handle->arrayBase=base;
```

```

(arrayStruct*) handle->length=length;
push handle;
}

```

5 10. The method according to claim 2 wherein constructing the object comprises:

defining a memory map having a predetermined address space for the hardware peripheral, and allocating at least one additional address space contiguous with the predetermined

10 address space;

storing object header information for the object directly in the additional address space;

creating an object handle for the object which points to the object header.

15

11. The method according to claim 10 further comprising:

defining a Java class having a base address parameter which is used in constructing said object.

20 12. The method according to claim 11 wherein the Java class is substantially defined in pseudocode as follows:

Class AnchoredArray

{

public int element[];

25 public AnchoredArray(int baseAddress)

{

element = lockDownElements(type,baseAddress);

Static private native int[] lockDownElements(int baseAddress);

}

30

Void AnchoredArray_lockdownElements()

{

int base = popStack();

```
pushStack(base);  
}
```

13. A device comprising a memory, wherein the memory
5 comprises elements defining a Java like object such that the
object overlaps with a predetermined address space of the
memory, the address space comprising a peripheral's memory
mapped registers.

10 14. A device comprising a memory, wherein the memory
comprises a class which enables an object to be defined such
that it overlaps with a predetermined address space of the
memory.

15 15. A device comprising a memory, wherein the memory
includes:
a Virtual Machine; and
a class which enables an object to be defined such
that it overlaps with a predetermined address space of said
20 memory.

16. A processor comprising:
a plurality of peripheral memory mapped registers;
an object anchored to said peripheral memory mapped
25 registers.

17. The processor according to claim 16 further
comprising a new system class having base address and length
parameters, which is used in constructing said object.

30

18. The processor of claim 16 further comprising a class
having type, base address and length parameters, which is used
in constructing said object.

19. The processor according to claim 16 further comprising a new class having a class name <class name> as follows:

<class name> (base, length)

5 where <class name> is the name assigned to the new class, base is a parameter which specifies a beginning address, and length is a parameter specifying a number of elements in the object, which when constructed, generates an object descriptor specifying base, length, and a generates a handle which points
10 to the object descriptor.

20. The processor according to claim 16 comprising:

a memory map having a predetermined address space for each of a plurality of peripherals, the memory map having
15 additional space for header information;

an object defined to overlap with the predetermined address space with a header stored in the additional space.

21. The processor according to claim 16 comprising:

20 a memory map having a predetermined address space for each of a plurality of peripherals;

for each of the plurality of peripherals, a Java object descriptor defined to point to the predetermined address space.

25

22. A method of providing application layer access to a fixed memory address space for an application in a language designed to prevent accessing particular memory locations directly, the method comprising:

30 constructing an object in the context of the language having elements which occupy said fixed memory address space;

whereby the application is provided access to the fixed memory address space directly through said object.

23. The method according to claim 22 further comprising:
identifying the fixed memory address space to be a
hardware peripheral's memory mapped registers.

5

24. The method according to claim 23 wherein constructing the
object comprises:

defining a memory map having a predetermined address
space for the hardware peripheral, and allocating at least one
10 additional address space contiguous with the predetermined
address space;

storing object header information for the object
directly in the additional address space; and

creating an object handle for the object which points
15 to the object header.

25. The method according to claim 22 further comprising:

defining a new class having a base address parameter
which is used in constructing said object.

20

26. The method according to claim 25 wherein the new class is
substantially defined in pseudocode as follows:

Class AnchoredArray

{

25 public int element[];

public AnchoredArray(int baseAddress)

```

    {
        element = lockDownElements(type,baseAddress);

        Static private native int[ ] lockDownElements(int
baseAddress);
5    }

Void AnchoredArray_lockdownElements()
{
    int base = popStack();
    pushStack(base);
10 }

```

27. The device according to claim 13, wherein the object is adapted for use in a Java-like programming environment.

15 28. The device according to claim 14, wherein the object is adapted for use in a Java-like programming environment.

29. The device according to claim 15, wherein the class is a Java class and said object is a Java object.

20